Santeri Qvintus

Turva-alueprototyypin paikannusjärjestelmä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

7.5.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Santeri Qvintus Turva-alueprototyypin paikannusjärjestelmä 31 sivua 7.5.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	Smart Systems and Software Engineering
Ohjaaja	yliopettaja Antti Piironen

Insinöörityön tarkoituksena oli rakentaa järjestelmä, joka voisi parantaa muistisairauksista kärsivien henkilöiden turvallisuutta. Työ tehtiin Lapinjärven kunnalle, jotta voitaisiin selvittää konseptin toimivuus.

Insinöörityössä selvitettiin nykyisiä valvontajärjestelmiä hoitokodeissa, mahdollisia vaaroja muistisairauksia sairastaville ja turvajärjestelmän vähimmäisvaatimukset. Työssä päädyttiin käyttämään Android-puhelinta paikantimena, CloudMQTT-palvelua laitteiden välisenä kommunikaatiokanavana, yhden piirilevyn Raspberry Pi -tietokonetta palvelimena ja Telegram-bottia kommunikaatioväylänä henkilölle.

Työssä käytettyjen laitteiden avulla saatiin rakennettua järjestelmä, joka pystyy valvomaan henkilön sijaintia, tunnistamaan, onko henkilö turva-alueen sisällä, kommunikoimaan henkilölle ja lähettämään viestin toisen opiskelijan opinnäytetyöhön kuuluneeseen drooniin. Toisen opiskelijan insinöörityössä tehty drooni lentää tässä työssä tehdyn palvelimen lähettämiin koordinaatteihin, eli henkilön koordinaatteihin, ja lähettää livekuvaa takaisin, jotta voidaan tarkistaa henkilön turvallisuus.

Insinöörityössä tehdyllä prototyypillä on mahdollista demonstroida konseptin toimivuus. Työn lopputuloksena luotiin onnistuneesti järjestelmä, joka täyttää vähimmäisvaatimusten kriteerit.

Avainsanat	Android, CloudMQTT, GPS, Raspberry Pi, Telegram, drooni

Author	Santeri Qvintus
Title	Positioning system of safe area prototype
Number of Pages	31 pages
Date	7 May 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Smart Systems and Software Engineering
Instructor	Antti Piironen, Principal Lecturer

The goal of the thesis was to create a system, that could improve the safety of people with memory impairments. This project was done for Lapinjärvi municipality so that proof of concept could be verified. The project was partly done with another student.

The current methods of security in the care homes, possible dangers for people with memory impairments and the minimum requirements for the system was investigated in this thesis. Android phone was used as a locator, CloudMQTT as device-to-device communications, Raspberry Pi the single-board computer as server and Telegram bot as way to send user messages in the project.

With these components a successful system was built, which could monitor the location of a person, recognize if the person is within the safe area, to communicate with the person and to send messages to the drone that was made by another student for another thesis. The drone made in another student's thesis would fly to the coordinates sent by the server of this thesis, so that the safety of the person could be verified by the video stream sent by the drone.

It is possible to demonstrate the proof of a concept with the prototype that was made in the thesis. The result of the thesis was a successful system, that fills all the minimum requirements.

Keywords	Android, CloudMQTT, GPS, Raspberry Pi, Telegram, drone

Sisällys

Lyhenteet

ADI	Alzheimer's Disease International. Alzheimerin tautia sairastavia henkilöitä auttava organisaatio.
IoT	Internet of Things. Esineiden internet, eli internetverkon laajentuminen laitteisiin.
JSON	JavaScript Object Notation. Avoimen standardin tiedostomuoto.
MQTT	Message Queuing Telemetry Transport. Julkaisuihin ja tilauksiin perustuva viestintäprotokolla.
UAV	Unmanned Aerial Vehicles. Lentävä miehittämätön ilma-alus.

1 Johdanto

Insinöörityön tarkoituksena oli kehittää järjestelmä, joka pystyy valvomaan henkilöiden koordinaatteja ja lähettämään komentoja muille järjestelmille. Ajatuksena oli kehittää henkilöiden turvallisuutta valvova järjestelmä, jonka voisi ottaa käyttöön henkilöiden valvonnassa. Työn tavoitteena oli valmistaa prototyyppi, jonka avulla voitaisiin todeta konseptin toimivuus. Tämä projekti tehtiin Lapinjärven kunnalle Metropolia Ammattikorkeakoulussa.

Henkilöiden koordinaatteja lähettäisi Android-puhelin, johon on asennettu sovellus. Android-sovellus lähettäisi puhelimen koordinaatit MQTT-palvelimelle. Palvelin käsittelisi MQTT-palvelimelle saapuvaa dataa ja tarkistaisi, onko henkilö ennalta määritellyn alueen sisä- vai ulkopuolella. Palvelin ylläpitäisi myös verkkosivua, jossa näytettäisiin Google-kartassa turva-alueen rajat sekä henkilön ja kopterin sijainnit. Projektissa käytettäisiin Raspberry Pi 3 B -tietokonetta Raspbian-käyttöjärjestelmällä palvelimena.

Tähän prototyyppiin liittyy myös toisen opiskelijan, Ville Sorkkilan, tekemä erillinen insinöörityö. Sen tarkoituksena olisi ohjelmoida drooni, joka voitaisiin lähettää tarkistamaan henkilön turvallisuus. Drooni odottaisi herätyskomentoa palvelimelta. Herätyksen jälkeen drooni lentäisi itsenäisesti henkilön koordinaatteihin ja lähettäisi livekuvaa päätelaitteeseen. Drooni palaisi takaisin alkuperäisiin koordinaatteihinsa kuvauksen jälkeen.

2 Hoitokotien turvallisuus

Vuonna 2013 maailmassa eli yli 35 miljoonaa jonkinlaista muistisairautta sairastavaa henkilöä. On odotettu muistisairautta sairastavien määrän nousevan kaksinkertaiseksi vuoteen 2030 mennessä ja kolminkertaistuvan vuoteen 2050 mennessä. Heistä monen kohtalo on huono. ADI (Alzheimer's Disease International) on julkaissut raportin, jonka tavoitteena on parantaa muistisairauksia sairastavien henkilöiden ja heidän läheistensä elämänlaatua. ADI:n mukaan noin puolet apua tarvitsevista vanhuksista sairastaa dementiaa ja hoitokodeissa asuvista jopa 80 prosenttia. ADI kehottaa hallituksia laatimaan pitkän tähtäimen suunnitelmia, miten muistisairauksia sairastavien henkilöiden pitkäaikaishoitoa tulevaisuudessa turvattaisiin. Myös dementiahoidon laadunvalvontaa laitoksissa ja kotona tulisi valvoa. [1.]

Yksi huomioitava tapaus on 84-vuotiaan Alzheimerin tautia sairastavan henkilön menehtyminen Helsingissä vuonna 2016. Tämä tapaus olisi voitu välttää käyttämällä tehokkaampia valvontamenetelmiä ja tekemällä järjestelmästä varmempi. Tapaus alkoi siitä, kun henkilö astui perjantai-iltana puoli yhdentoista aikoihin ulos asunnostaan. Ulko-ovissa oli sensorit, jotka lähettävät Palmia-valvontayritykselle viestin oven avauksesta. Hälytykset ovat olleet ympärivuorokautisia, mutta tässä tapauksessa päivähälytykset oli otettu pois päältä, sillä Palmian mukaan henkilö oli aiheuttanut liikaa hälytyksiä. Vasta seuraavana aamuna kaupungin kotiavustaja saapui asunnolle tavalliseen tapaan, mutta henkilöä ei löytynyt asunnosta. Kotiavustaja soitti hätäkeskukseen, ja poliisit saapuivat paikalle. Alzheimeria sairastavalle henkilölle oli myös annettu hälytysranneke, mutta ranneke oli jätetty kotiin. Tapaus päättyi siihen, että henkilö löydettiin usean päivän jälkeen menehtyneenä pakkasesta noin sadan metrin päästä asunnostaan. Jos tässä tapauksessa olisi käytetty varmempaa paikannusjärjestelmää, olisi henkilö voitu löytää tarpeeksi ajoissa tai koko tapaus välttää. [2.]

Muistisairauksia sairastavilla henkilöillä on arjenvietto vaikeampaa kuin sairastamattomilla. Esimerkiksi äsken puhuttujen tai sovittujen asioitten muistaminen voi olla vaikeaa. Ohjeitten mukaan toimiminen on myös vaikeaa. Ajan ja paikan taju hämärtyy sairastumisen myötä. Muistisairauksia sairastava henkilö voi eksyä niin vieraassa kuin tutussa-

kin ympäristössä ja unohtaa, mihin oli menossa. Myös yön ja päivän erottaminen toisistaan vaikeutuu, samoin eri vuodenaikojen tunnistaminen. Tästä syystä muistisaurauksia sairastavilla henkilöillä voi olla vääränlainen vaatetus ulkoillessa. [3.]

Monet muistisairaat ovat ikäihmisiä, sillä muistisairaudet, kuten dementia, tulevat usein iän myötä. Ikäihmiset kuuluvat myös helleajan riskiryhmään. Yli 25 lämpöasteen jälkeen alkaa ilmetä kuumasairauksia, joista voi seurata uupumusta, kuumakutinaa, turvotusta, pyörrytystä, kouristuksia ja lämpöhalvauksia. Vanhukset ja sydän- ja verisuonitauteja sairastavat henkilöt ovat erityisen riskialttiita näille. Tutkimusten mukaan kuumuus Suomessa voi aiheuttaa 180–900 kuolemaa joka vuosi [4.]. Myös pakkanen voi olla vaaraksi, varsinkin jos henkilö on joutunut lukkojen taakse ulos pakkasella. Muistisairauksista kärsivät henkilöt voivat helposti unohtaa avaimensa tai muita samantyyppisiä esineitä sisälle. Tästä syystä olisi hyvä, jos henkilöillä olisi mukana laite, joka pystyy paikantamaan ja mittaamaan lämpötilan. Jotta muistisairaudesta kärsivä henkilö muistaisi ottaa laitteen mukaansa, olisi laitteen oltava kiinni jossain, mitä henkilö tyypillisesti kantaa aina mukanaan.

3 Uuden turvallisuusjärjestelmän tavoitteet

Insinöörityönä tehdyn turva-alueprototyypin tavoitteena oli luoda järjestelmä, joka sisältää kaikki tarvittavat komponentit vähimmäistuotteeseen. Tavoitteiden lisäksi oli myös asetettu ylimääräisiä toimintoja, joita voitiin tehdä, jos ylimääräistä aikaa jäisi jäljelle prototyypin vähimmäistavoitteiden saavuttamisen jälkeen. Ylimääräisiä toimintoja alettaisiin kehittää vasta, kun tarvittavat komponentit toimivat tehokkaasti ja luotettavasti. Järjestelmän tulisi olla valmistuttuaan niin luotettava, että sen voi jättää toimimaan ympärivuorokautisesti.

3.1 Henkilön paikannus

Jotta henkilöiden sijaintia voidaan paikantaa, on käytettävä GPS (Global Positioning System) -järjestelmää hyödyksi projektissa. GPS paikantaa laitteen pituus- ja leveysasteen koordinaatit. Näiden koordinaattien avulla voidaan laskea henkilön matka turva-alueen keskipisteestä. Koordinaatit vaihtelevat jonkin verran, joten liian tarkkaa järjestel-

mästä ei kannata tehdä. On kehitettävä laite, joka toimii paikantimena ja lähettää koordinaatit palvelimelle viestitysprotokollan avulla. Palvelin vertaa henkilön koordinaatteja ja laskee, kuinka kaukana henkilö on turva-alueen keskipisteestä.

Palvelimeen voidaan myös asentaa HTTP (Hypertext Transfer Protocol) -palvelin. HTTP-palvelin ylläpitäisi verkkosivua, josta voitaisiin nähdä henkilön ja droonin koordinaatit sekä alueen rajat. Tähän tulisi käyttää valmiita karttoja, kuten esimerkiksi Google-karttaa. Kartan avulla nähtäisiin helposti suhteellisen tarkat sijainnit jokaisella laitteella. On myös mahdollista kehittää työkaluja karttaan, josta voitaisiin asettaa uusia alueita turva-alueeksi. Tämän kehittäminen vaatisi paljon aikaa, joten prototyypissä on parempi käyttää väliaikaista turva-alueen asettamista paikantimen avulla.

3.2 Kommunikaatio laitteiden välillä

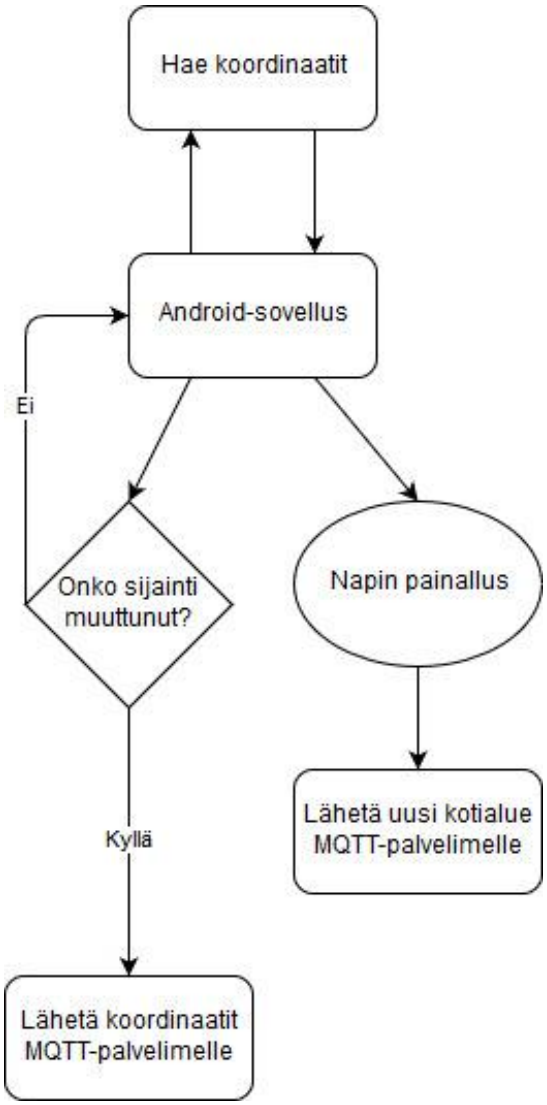
Laitteiden välinen kommunikaatio on tärkeä järjestelmän kannalta, sillä jokainen laite suorittaa tietyn tehtävän prototyypissä. Viestin tyyppi on hyvin datamaista, kuten koordinaatteja tai aktivointiviestejä. Data voidaan tehdä laiteystävälliseksi käyttämällä yleistä JSON (JavaScript Object Notation) -datamuotoa. Näin voidaan käyttää valmiita JSON-paketoijia ja -purkajia. Välittäjän tulisi vastaanottaa viestejä, vaikka kaikki laitteet eivät olisi päällä. Tämä tekee järjestelmien testaamisesta helpompaa, kun kaikkien järjestelmien ei tarvitse olla päällä.

Palvelimen täytyy myös lähettää viestejä henkilölle. Tähän tarvitaan jonkinlainen kommunikaatioväylä laitteen ja ihmisen välillä. Tarjolla on monia keskustelupalveluita, joihin voidaan tehdä viestipääte bottina. Palvelin lähettää viestejä botille ja botti näyttää viestit käyttäjälle. Botin täytyy pystyä vastaanottamaan viestejä välittömästi, jotta henkilö tietää heti, milloin hän astuu alueen ulkopuolelle.

4 Turvajärjestelmän toteutus

4.1 Paikannusjärjestelmä

Henkilöiden sijainnin paikantamiseen käytettiin Huawei Honor 7 -Android-puhelinta. Paikannusjärjestelmän tehtävä on lähettää laitetta kantavan henkilön liikkuessa laitteen koordinaatit palvelimelle. Paikannussovellus voi myös asettaa uuden turva-alueen, jos käyttäjä painaa ”set home” -nappia. Napin painallus asettaa turva-alueen keskipisteen henkilön sijainnin kohdalle. Turva-alue on prototyypissä aina ympyrän muotoinen, joten sovelluksessa voi asettaa myös halkaisijan pituuden metreinä. Kuvassa 1 näkyy kaavio paikannusjärjestelmän toimintaperiaatteesta.



Kuva 1. Insinöörityössä tehdyn Android-sovelluksen toimintaperiaate.

Prototyypissä käytetty Huawei Honor 7 -puhelin oli helppo valinta, sillä tämä puhelin oli tekijän henkilökohtainen puhelin. Puhelimessa oli riittävän uusi Android-versio, jotta kaikki tarvittavat toiminnot olivat saatavilla. Moderneissa puhelimissa on esimerkiksi sisäänrakennettu GPS ja mobiiliverkko. Nämä ominaisuudet olivat tärkeitä, sillä puhelimen päätarkoitus oli lähettää omat koordinaattinsa verkon kautta MQTT (Message Queuing Telemetry Transport) -palvelimelle. Tähän tarvittiin mobiiliverkkoa, sillä paikannusjärjestelmän tuli toimia Wi-Fi-verkoista riippumatta.

Sovellus ohjelmoitiin Googlen omalla Android Studiolla käyttäen Java-kieltä, ja ulkoasut määrittää XML-tiedostot. Prototyypissä käytettyyn Android-kehitysympäristön projektiin tehtiin kaksi erillistä luokkaa, MainActivity ja MqttHelper. MainActivity on ensimmäinen luokka, joka käynnistyy sovellus käynnistyessä. Tässä luokassa sovellus käynnistää GPS-paikannuksen ja MqttHelper-luokan. Main-luokka myös tarkistaa, onko sovelluksella tarvittavat oikeudet, ja tarvittaessa kysyy käyttäjältä lupia.

MqttHelper-luokka nimensä mukaisesti hoitaa sovelluksen MQTT-yhteydet. Luokkaan on kirjattu CloudMQTT-palvelimelle tarvittavat tiedot, kuten URL (Uniform Resource Locator), käyttäjänimi, salasana ja viestien merkinnät. Luokka pyrkii ottamaan yhteyden palvelimeen ja jos tämä onnistuu ongelmitta, on yhteys palvelimelle luotu. Tarvittaessa luokassa olevia ”publishToTopic” -funktioita voidaan kutsua. Nämä luokat hoitavat viestin lähetyksen CloudMQTT-palvelimelle.

Kun sovellus lähettää henkilön koordinaatit tai uuden turva-alueen sijainnin, se käyttää MQTT-protokollaa viestittämisväylänä palvelimelle. Androidissa käytetty Eclipse Paho on vuonna 1998 alkunsa saanut kevyt viestitysprotokolla, jonka alkuperäiset tekijät ovat olleet IBM (International Business Machines Corporation) ja Arcom, mutta myöhemmin se on siirtynyt Eurotechin omistukseen. [5.]

Jotta Android-sovelluksessa pystyttäisiin käyttämään Paho MQTT -kirjastoa, se on lisättävä projektin gradle-tiedostoihin. Android-sovelluksessa on kaksi tiedostoa, joihin on lisättävä viittaukset Paho MQTT -kirjastoon, projektin gradle ja sovelluksen gradle. Kuvassa 2 näkyvät tarvittavat tiedot, jotka tulee lisätä ”repositories” -kohtaan. [6.]

```
buildscript {

    repositories {
        google()
        jcenter()
        maven {
            url "https://repo.eclipse.org/content/repositories/paho-snapshots/"
        }
    }
```

Kuva 2. Lisättävä rivi projektin gradle-tiedostossa.

Paho MQTT kuitenkin tarvitsee lisäyksiä toiseen gradle-tiedostoon. Toinen gradle-tiedosto on sovelluksen oma gradle-tiedosto. [6.] Tähän tiedostoon tarvitaan kaksi koodiriviä, jotka näkyvät kuvassa 3.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
    compile 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.0'
    compile 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
}
```

Kuva 3. Lisättävät rivit sovelluksen gradle-tiedostossa.

Paho MQTT -kirjaston lisäämisen lisäksi on Android-projektissa merkittävä se, että sovellus pyytää lupaa käyttää puhelimen erilaisia ominaisuuksia, kuten verkon käyttöä ja GPS-paikantamista. Nämä rivit on lisättävä AndroidManifest-tiedostoon. Kuvassa 4 näkyvät koodirivit jotka tulee lisätä, jotta sovellus toimii oikein.

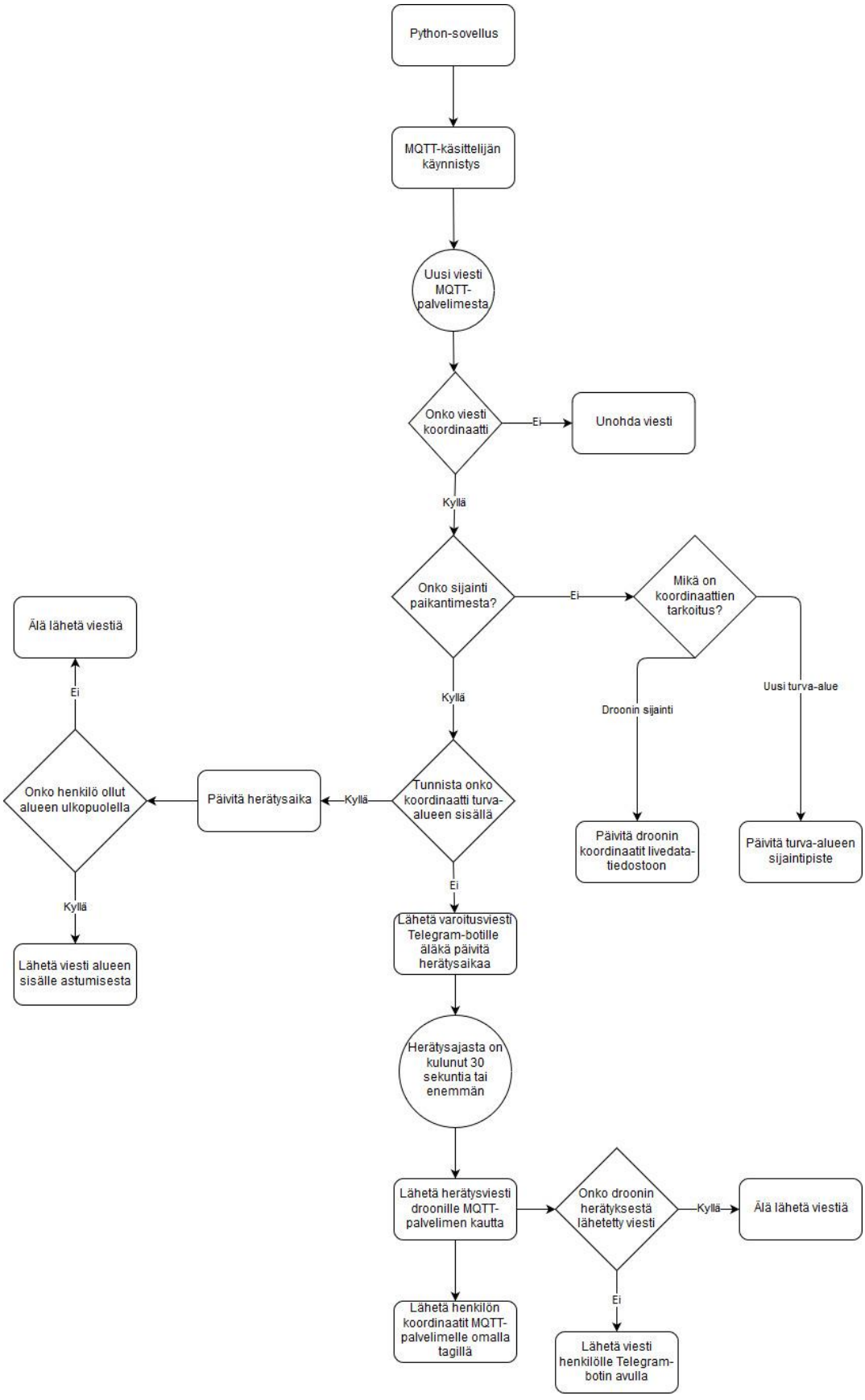
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<uses-feature android:name="android.hardware.location.gps" />
```

Kuva 4. Manifestitiedostoon lisättävät koodirivit.

4.2 Palvelin

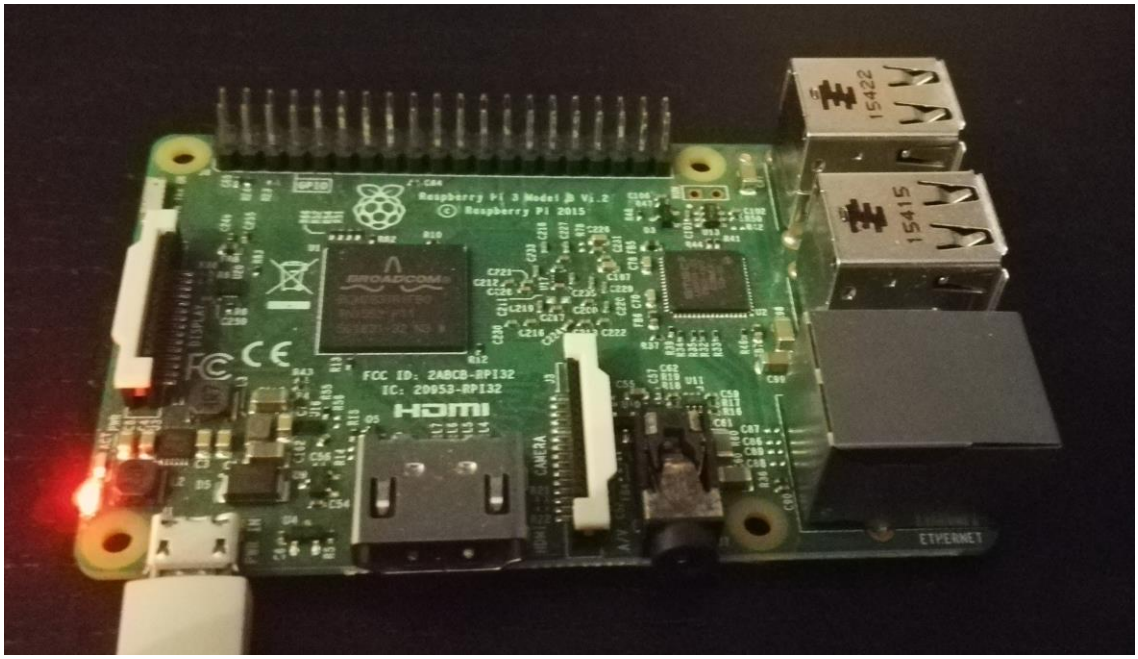
Palvelin on tietokone, joka on suunniteltu prosessoimaan pyyntöjä ja lähettämään dataa toisiin koneisiin. Usein ajatellaan palvelimia HTTP-palvelimina, ja nämä HTTP-palvelimet ylläpitävät verkkosivuja. On kuitenkin monia erilaisia palvelimia ja näillä palvelimilla on erilaisia käyttötarkoituksia. Tarkalleen ottaen palvelin on ohjelmisto, joka ylläpitää tiettyä palvelua, mutta usein tehokkaat laitteet lasketaan myös palvelimiksi, sillä tähän tarkoitukseen käytetyt laitteet ovat usein tehokkaampia, kuin mitä kuluttajat ostavat itselleen. [7.]

Palvelimen tarkoitus on toimia tiedon prosessointi- ja välityslaitteena. Insinöörityön prototyypissä palvelin vastaanottaa erilaisia viestejä ja suodattaa turhat viestit pois selvittämällä onko viesti koordinaatti vai ei. Tämän jälkeen palvelin selvittää, onko viesti paikantimesta vai muualta. Jos viesti on tullut muualta, palvelin selvittää, ovatko koordinaatit droonin sijainti vai uusi turva-alue, ja päivittää tiedot. Jos koordinaatit ovat peräisin paikantimesta ja ne ovat henkilön koordinaatteja, palvelin laskee tulevatko koordinaatit alueen sisältä vai ulkopuolelta. Jos henkilö on astunut turva-alueen ulkopuolelle, lähetetään viesti rajan ylityksestä Telegram-botille ja merkitään herätysaika muistiin. Kun on kulunut 30 sekuntia ja henkilö on vieläkin alueen ulkopuolella, lähettää palvelin herätysviestin droonille ja vielä toisen viestin henkilölle tästä. Jos henkilö astuu tämän jälkeen alueen sisälle, ei droonia tarvitse herättää ja henkilölle lähetetään viesti, että hän on palannut takaisin alueen sisälle. Palvelimen toimintaperiaatetta havainnollistaa kuva 5.



Kuva 5. Raspberry Pi -palvelimen toimintaperiaate.

Palvelimeksi valittiin Raspberry Pi 3 B -mallin yhden piirin tietokone. Raspberry Pihin asennettiin Linux-pohjainen Raspbian-käyttöjärjestelmä. Linux-pohjaiset käyttöjärjestelmät sopivat hyvin tällaisiin käyttötarkoituksiin, sillä ne ovat ilmaisia, nopeasti kehittyviä ja alustariippumattomia kehitysalustoja. [8, s 4.] Raspberry Pissä on sisäänrakennettu langaton verkkoyhteys ja Bluetooth. Prosessorina käytössä on 64-bittinen ARM Cortex-A53 -neliydinprosessori. Laitteessa on myös Broadcoming Videocore 4 -näytönohjain. Laitteeseen suositellaan virtalähdettä, joka pystyy syöttämään jatkuvasti vähintään 2,5 ampeeria ja 5.1 volttia virtaa [9, s. 12–13.]. Kuvassa 6 on projektissa käytetty Raspberry Pi 3 B.



Kuva 6. Projektissa käytetty Raspberry Pi -tietokone.

Raspberry Pihin kuuluu myös 40 pinniä, joista 26 on GPIO (General Purpose Input/Output) -pinnejä. Pinnien avulla on Raspberry Pihin mahdollista liittää erilaisia sensoreita ja muita elektroniikkalaitteita. [9, s. 13–14.] Jos palvelin sijoitettaisiin turva-alueen sisälle tai lähistölle, siihen olisi mahdollista kiinnittää sensoreita, joilla esimerkiksi voisi lähettää henkilöille paikallisen sään tai pienentää turva-aluetta huonon sään aikana. Raspberry Pi 3 B -mallissa on myös neljä USB 2.0 -porttia, LAN-portti, DSI Display-portti, CSI Camera-portti, HDMI-portti ja neljä Pole Stereo Output and Composite Video -porttia [9. s 13.] Raspberry Pi valittiin projektiin, koska se on joustava kehitysalusta, edullinen ja

energiatehokas. Näiden ominaisuuksien lisäksi Raspberry Pi oli valmiiksi saatavilla projektia varten. Raspberry Pitä voi palvelimen lisäksi käyttää moniin eri käyttötarkoituksiin. Se sopii erityisesti tee se itse -projekteihin. [10.]

Palvelimen Python-ohjelma käynnistää MQTT:n, sillä kaikki ohjelman toiminnot riippuvat datan vastaanottamisesta. Viestin tullessa ohjelmisto lukee viestissä olevat ensimmäiset merkit. Nämä merkit kertovat ohjelmalle viestin lähteen. Suurin merkitys on koordinaattien ja uuden turva-alueen vastaanottamisessa. Tästä syystä uuden alueen määrittämisessä ohjelma tulostaa erilaisen tulostuksen kuin koordinaattien kanssa.

Jos ohjelmaan saapuvat koordinaatit ovat peräisin henkilöltä, ohjelma vertaa koordinaatteja turva-alueeseen ja laskee, onko henkilö alueen sisällä vai sen ulkopuolella. Ohjelma myös ylläpitää hälytysaikaa, joka on otettu henkilön vielä ollessa alueen sisällä. Jos henkilö astuu alueen ulkopuolelle, verrataan nykyistä aikaa herätysaikaan. Ohjelma myös tulostaa tietoa kuten pituus- ja leveysaste, turva-alueen halkaisija, sijainnin pituus- ja leveysaste ja onko drooni herätetty. Jos henkilö on alueen ulkopuolella, tulostaa ohjelma myös nykyisen ajan ja herätysajan. Esimerkki tulostuksista näkyy kuvassa 7.

```
Values received:
source: p1
latitude: 60.22108481240792
longitude: 24.80477330495339
safe are lat: 60.2203346781
safe area lon: 24.8039156481
safe area size: 0.005
droneWoken: False
current time: 2018-04-09 11:43:55.900596
alert time: 2018-04-09 11:44:05.600993
coordinates sent
```

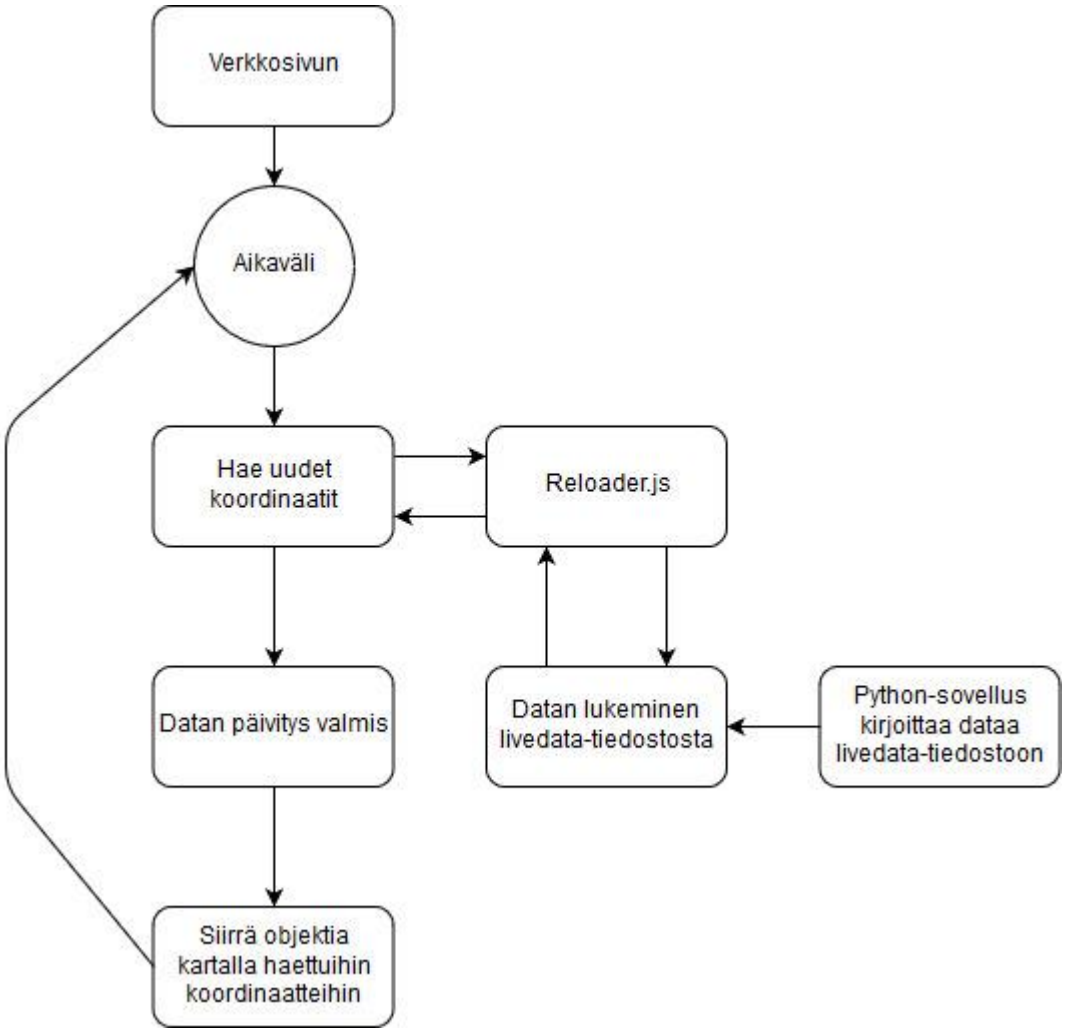
Kuva 7. Python-tulostukset, kun palvelin vastaanottaa uuden koordinaatin.

Droonin tulostukset ovat lyhyempiä kuin henkilön, sillä droonin koordinaatteja ei verrata mihinkään alueeseen. Droonin koordinaattien ainut tarkoitus on merkitä merkki Google-karttaan verkkosivulla. Ohjelmaan on myös tehty virheenkäsittelyä, jos tietyissä koodinosissa tulee virhe. Virheenkäsittely toimii sijoittamalla tarvittavat koodinosat ”try”-osion sisälle. Jos virhe tulee ”try”-osion sisällä, menee ohjelma ”catch”-osion sisälle. Tämä estää ohjelman täydellisen kaatumisen ja sen sijaan vain tietty osa kaatuu. Ohjelma tulostaa konsoliin, että virhe on tapahtunut.

4.2.1 Verkkosivu

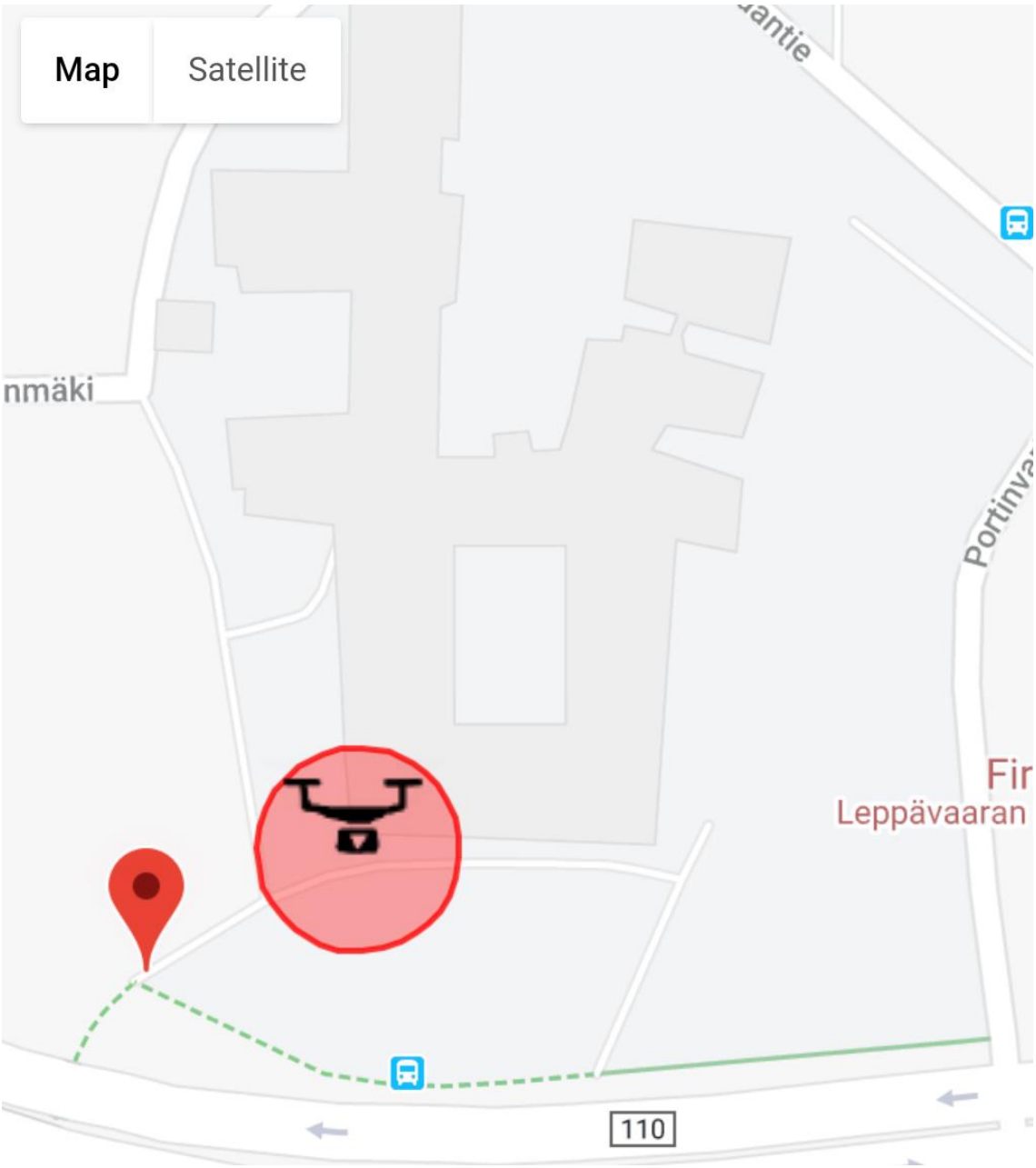
Turvajärjestelmän verkkosivun tarkoitus on näyttää ylläpitäjälle tai käyttäjälle henkilön, droonin ja turva-alueen sijainnit. Tähän tarkoitukseen käytettiin Googlen kehittämää karttaa, jota voi käyttää omalla sivulla. Sivu käyttää toimiakseen kolmea eri tiedostoa ja kartassa käytettävää kuvaa, jolla merkitään droonin sijainti. ”Reloader.js” on JavaScript-tiedosto, jonka tarkoitus on lukea livedata-tiedostoa. Livedata-tiedostoon on kirjoitettu henkilön ja droonin koordinaatit sekä turva-alueen sijainti. Verkkosivu käyttää myös PHP (Hypertext Preprocessor) -tiedostoa. Tämä tiedosto kokoaa verkkosivun ja Google kartan. Turva-alueen piirtämisen ja merkkien päivitykseen käytettiin ajoitettuja funktiokutsuja. Jos näitä funktiokutsuja ei olisi, pitäisi käyttäjän itse päivittää sivu saadakseen merkit ja alueen päivitettyä.

Projektissa käytettiin Apache-nimistä HTTP-palvelinta. Apache-palvelin on yhteistyössä tehty luomaan tehokas, laadukas, ilmainen avoimen lähdekoodin toteutus HTTP-palvelimesta. Apachea kehittävät kehittäjät ympäri maailmaa vapaaehtoisesti. [11.] Vuonna 2006 noin 70 % palvelimista oli Apache-palvelimia, ja suurin osa näiden palvelimien alustoista oli UNIX-pohjaisia. [12, s 43.] Apachen saa asennettua Raspbian-käyttöjärjestelmään nopeasti. Asennus alkaa, kun konsoliin on syötetty ”sudo apt-get install apache2 -y” -komento. Ennen asentamista käyttöjärjestelmä tulee olla päivitettynä ”sudo apt-get update”- ja ”sudo apt-get dist-upgrade” -komennoilla. Oletuksena Apache testi HTML (Hypertext Markup Language) -tiedoston palvelimen oletussivuksi. Apache on asennettu onnistuneesti, jos oletussivu näkyy, kun ”http://localhost/” kirjoitetaan internetselaimeen. [13] Verkkosivun toimintaperiaate näkyy kuvassa 8.



Kuva 8. Turvajärjestelmän verkkosivun toimintaperiaate.

Kartassa henkilöt on merkitty valmiiksi tehdyillä merkeillä. Drooneihin on projektia varten piirretty oma siluettimerkkikuva droonista merkitsemään droonin sijainti kartalla. Turva-alue näkyy kartalla punaisena alueena. Turva-alue päivittyy merkkejä hitaammin, sillä päivityksen aikana alue poistetaan ja sijoitetaan uudelleen. Jos aluetta päivitettäisiin esimerkiksi jokainen sekunti, voisi liian nopea päivittäminen rasittaa silmiä. On myös mahdollista vaihtaa karttakuva satelliittikuvaan, jossa kartta näkyy maisemakuvana. Kuvassa 9 näkyvät molemmat merkit ja turva-alue.



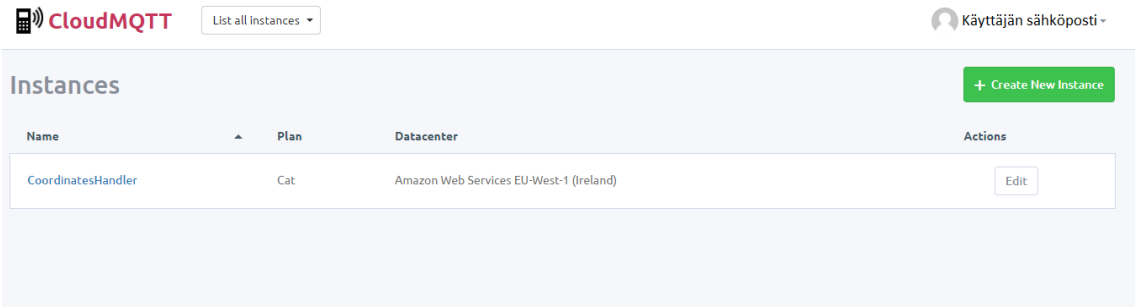
Kuva 9. Palvelimen ylläpitämän sivun kartta.

Kuva 9 on otettu testin aikana, ja se havainnollistaa, miltä Google-kartta näyttää, kun henkilö on kävellyt noin 10 metrin päähän turva-alueesta. Tässä tapauksessa drooni ei ollut lähtenyt vielä liikkeelle. Testien aikana huomattiin, että joissain tietyissä paikoissa saattaa olla huonompi GPS-saatavuus kuin toisissa. Järjestelmä toimii myös paikoissa, joissa on huono GPS-yhteys. On pidettävä mielessä, että tällaisissa paikoissa saattavat koordinaatit poiketa muutaman metrin enemmän verrattuna tavalliseen tarkkuuden vaihteluun. Koordinaattien tarkkuuden vaihtelu voi olla ohimenevää.

4.2.2 MQTT-protokolla

Insinöörityössä käytettiin CloudMQTT-nimistä maksutonta MQTT-palvelua. CloudMQTT käyttää pilvessä olevia Mosquitto-palvelimia. Mosquitto-palvelimet käyttävät MQTT-protokollaa, joka tarjoaa kevyen tavan välittää ja vastaanottaa viestejä. MQTT sopii hyvin esineiden internetiin (IoT, Internet of Things), jossa laitteet ovat yhteydessä internetiin. CloudMQTT:n tarjoaman palvelun ansiosta projektissa ei tarvinnut tehdä omaa MQTT-välittäjää. [14.]

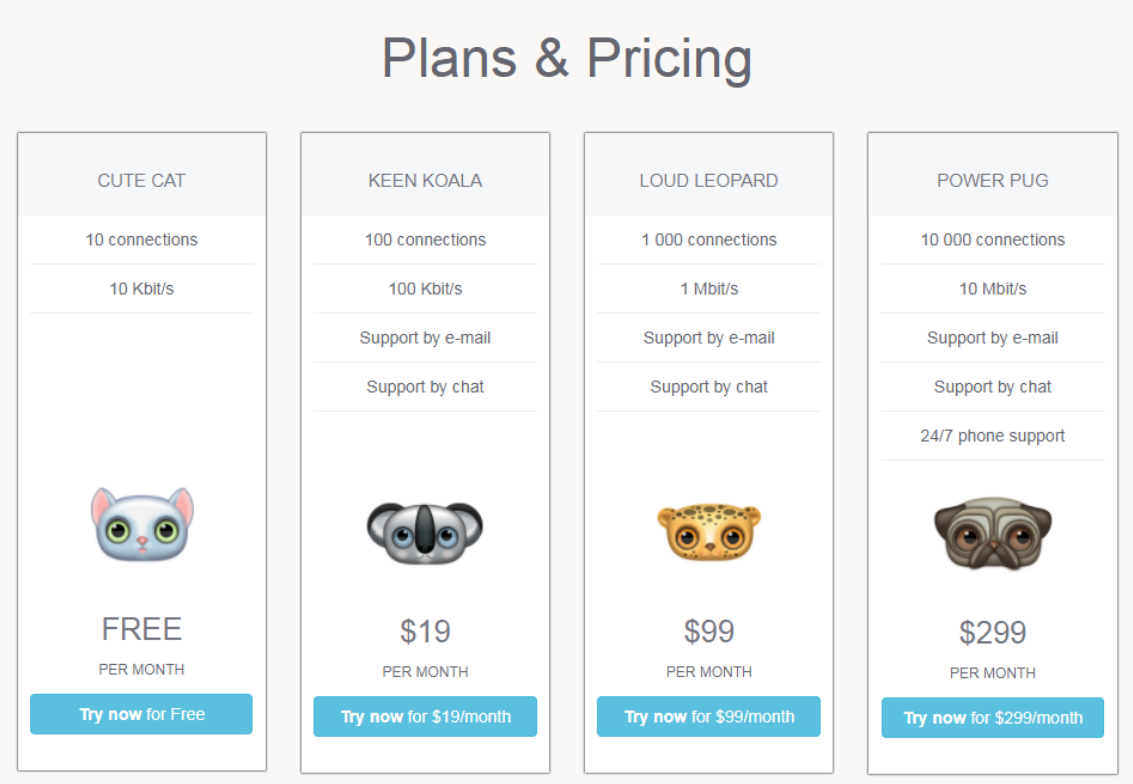
Jotta CloudMQTT-palvelua voi käyttää, on tehtävä tunnus tai kirjauduttava sisään GitHub- tai Google-tunnuksella. Kirjautumisen jälkeen käyttäjän on tehtävä uusi instanssi. Kuvassa 10 näkyy CloudMQTT:n instanssien luonti-ikkuna. Tässä tapauksessa on luotu vain yksi instanssi luotu, joka hallitsee kaikkia prototyyppiin kuuluvia viestejä. Uudelle instanssille on annettava nimi ja datakeskus, joka ylläpitää instanssia. Kuvassa 10 oleva datakeskus sijaitsee Irlannissa. [14.]



Kuva 10. CloudMQTT:n instanssit [14].

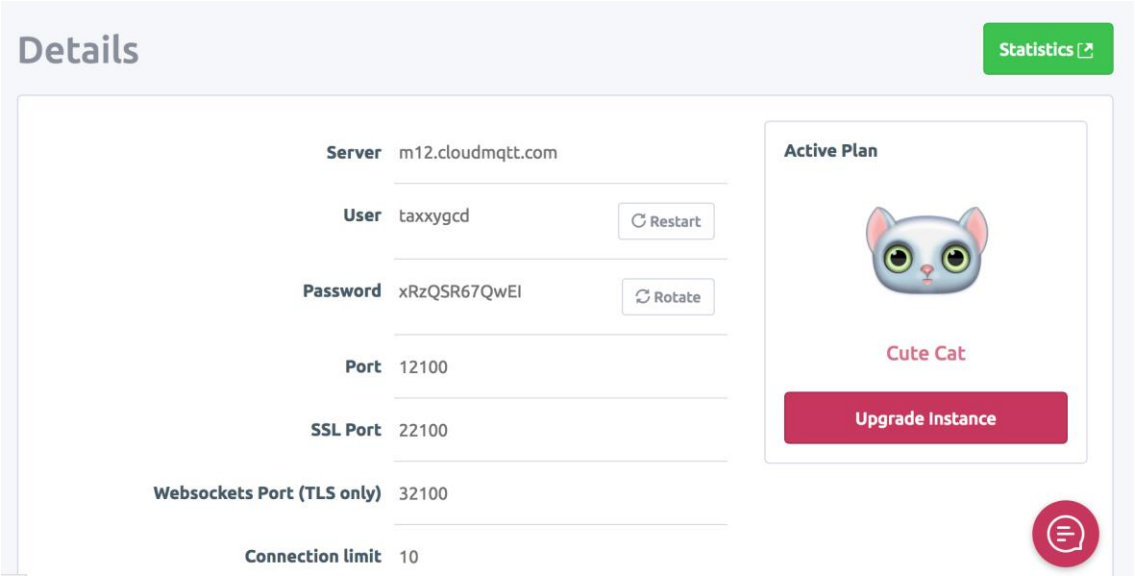
Kun käyttäjä luo uuden instanssin, hänelle avautuu uusi ikkuna, jossa valitaan nimi, suunnitelma, datakeskus ja mahdolliset tagit. Nimen voi valita instanssille vapaasti, mutta olisi suotavaa, että nimi antaisi kuvauksen instanssin käytöstä. Suunnitelman voi valita tarpeen mukaan. ”Cute Cat” on suunnitelma, joka soveltuu hyvin palvelun testaukseen. Tämä suunnitelma on ilmainen ja voi ylläpitää kymmentä eri yhteyttä. CloudMQTT-palvelussa on kolme muuta maksullista suunnitelmaa. Halvin maksullinen suunnitelma on ”Keen Koala”. Tämä suunnitelma maksaisi 19 dollaria kuukaudessa. Koala tarjoaa 100 eri yhteyttä laitteisiin, suuremman kaistanvaraukset, sähköposti- ja Chat tukipalvelut. ”Loud Leopard” -suunnitelma maksaisi 99 dollaria kuukaudessa. Tällä suunnitelmalla on samat ominaisuudet kuin Koalalla, mutta se tukee 1 000:ta yhteyttä ja suuremman kaistanvarauksen. Kallein suunnitelmista, ”Power Pug”, maksaisi 299 dollaria kuukaudessa.

Tässä suunnitelmassa on 10 000 yhteydelle tuki ja 10 megabittiä/sekunti. Sähköposti- ja chattituen lisäksi se tarjoaa ympärivuorokautisen puhelintukipalvelun. [15.] Eri suunnitelmat näkyvät kuvassa 11.



Kuva 11. CloudMQTT-palvelun tarjoamat suunnitelmat. [15].

Käyttäjä tarvitsee instanssin yksityiskohdista löytyvät palvelimen, käyttäjän, salasanan ja portin, jotta instanssiin voidaan lähettää viestejä. Ne löytyvät instanssin nimeä klikkaamalla. ”Details”- eli yksityiskohdat-välilehdestä löytyvät kaikki tarvittavat tiedot. Android-sovelluksessa otetaan yhteys palvelimeen laittamalla palvelimen merkkijono ja asettamalla portin luku merkkijonon perään kaksoispisteitten jälkeen. [14.] Kuva 12 näyttää, miltä instanssien yksityiskohdat -ikkuna näyttää.



Kuva 12. Instanssien yksityiskohdat. [14].

Yksityiskohtien lisäksi tärkeä välilehti instansseissa on ”Websocket UI”. Tästä välilehdestä käyttäjä voi nähdä kaikki viestit, jotka saapuvat instanssiin. Websocket UI -välilehdestä on myös mahdollista lähettää viestejä manuaalisesti. Viestien lähetykseen käyttäjän tarvitsee kirjoittaa aihe ja viesti. Manuaalisesti lähetetyillä viesteillä on helppo testata muitten järjestelmien toimivuutta ilman, että kaikkia laitteita tarvitsisi käynnistää kohdetestausta varten. Välilehdestä on myös mahdollista poistaa tietyn viestinlähettäjän lähettämiä viestejä. [14.]

Kuva 13 havainnollistaa, kuinka MQTT-protokollan viestitys toimii. Prototyypissä lähettäjä on paikannuslaite tai drooni, joka lähettää koordinaattinsa välittäjälle (cloudMQTT), palvelin tilaa välittäjältä saapuvia viestejä tietyn väliajoin ja välittäjä vastaa. CloudMQTT vaatii käyttäjän tekemisen, mutta haluttaessa voi käyttää Google-tunnusta. Jotta välittäjälle voi lähettää viestejä, tarvitaan kirjautumisnimeä ja salasanaa. [14.]

Kuva 13. Julkaisu-tilausviestitysmalli. [14].

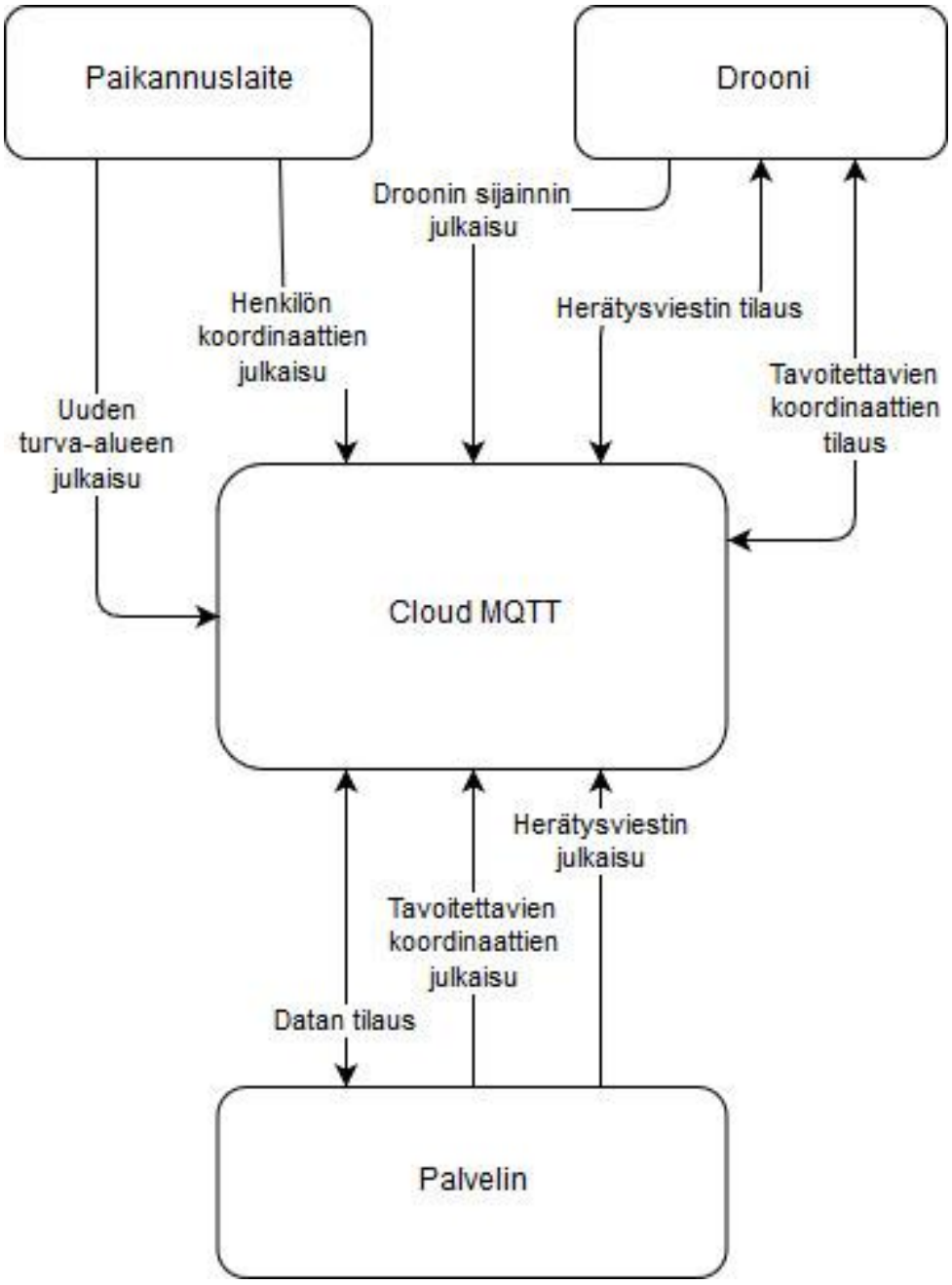
MQTT tukee myös erilaisia palvelun laatuja. MQTT-protokollassa on olemassa kolme eri vaihtoehtoa palvelunlaadulle. ”at most once”, ”at least once” ja ”exactly once”. HTTP-protokollassa ei ole samantyyppisiä palvelunlaatuja. Projektissa käytettiin ainoastaan “at most once” -palvelunlaatua. [16.]

- ”At most once” (enintään kerran) -palvelunlaatu takaa, että viesti lähetetään enintään vain kerran ja käytetään vähiten vaivaa [16].

- ”At least once” (ainakin kerran) -palvelunlaadulla viesti on vastaanotettu ainakin yhden kerran. Tässä varmistetaan, että viesti on saapunut perille, mutta viesti saattaa saapua monta kertaa välittäjälle [16].

- ”Exactly once” (tasan kerran) -palvelunlaadulla varmistetaan, että viesti saapuu tasan kerran [16].

Kuva 14 selventää, kuinka jokainen laite julkaisee ja hakee tietoa CloudMQTT-palvelusta. Paikannuslaitteella ei ole tarvetta tilata tietoa, mutta laite julkaisee usein henkilön koordinaatit. Drooni odottaa herätysviestiä palvelimelta, ja kun herätysviesti on saapunut, drooni tilaa henkilön koordinaatit ja aloittaa omien koordinaattiensa julkaisun. Palvelin hakee kaiken datan, johon on laitettu ”coordinates” eli koordinaattitagi. Tämän tagin ansiosta palvelin vastaanottaa vain koordinaatteja. Palvelin aloittaa koordinaattien julkaisun, jos henkilö on astunut alueen ulkopuolelle, mutta lähettää herätysviestin vasta, kun henkilö on ollut tarpeeksi pitkään alueen ulkopuolella.



Kuva 14. Laitteiden julkaisut ja tilaukset.

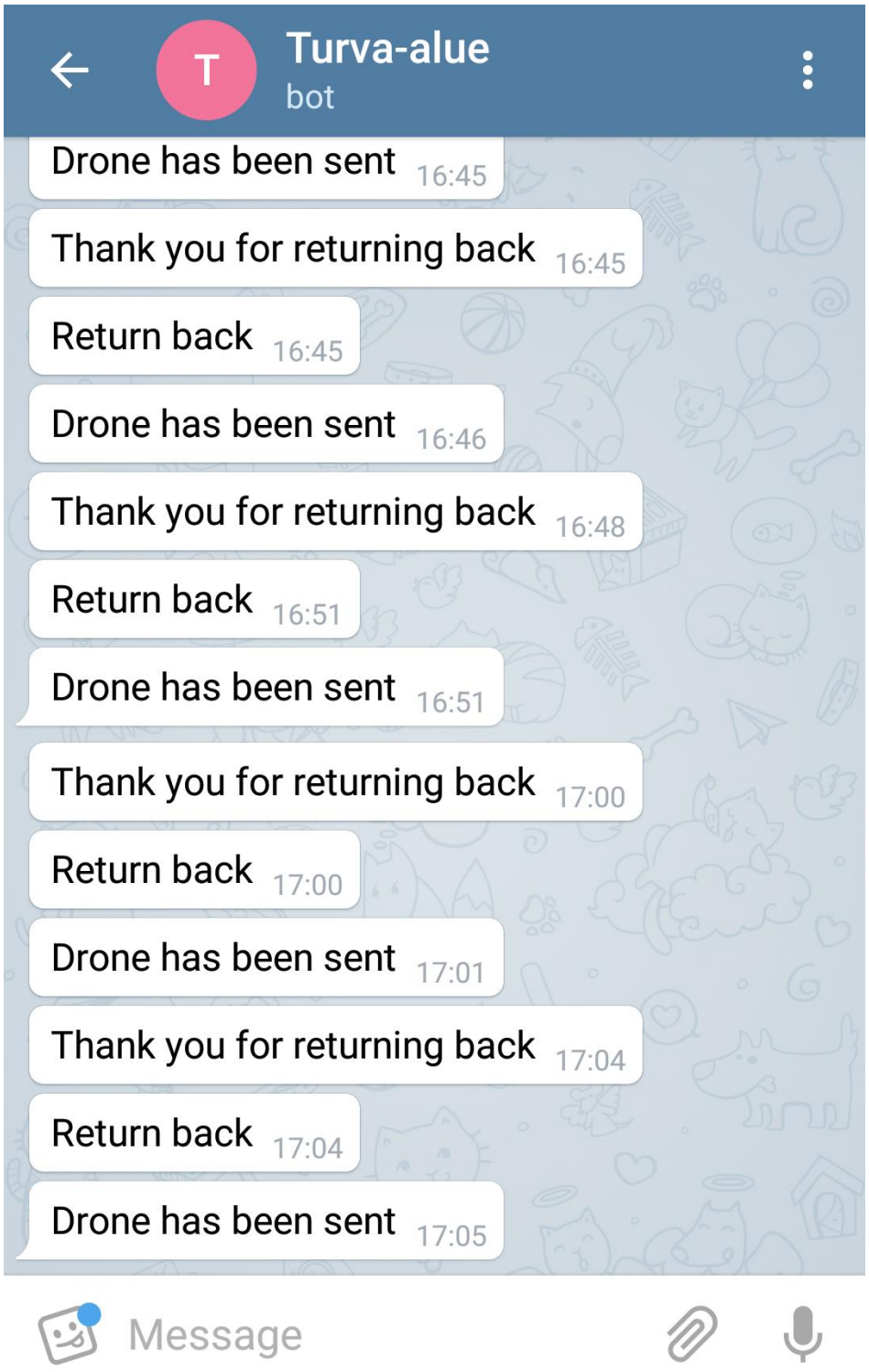
CloudMQTT valittiin projektiin maksuttomuuden, valmiin välittäjän, kommunikaation turvallisuuden ja yksinkertaisen datan välityksen takia. Myös HTTP-protokollaa harkittiin projektiin, mutta MQTT soveltuu paremmin datan lähetykseen ja hakemiseen. MQTT on myös kevyempi kuin HTTP-protokolla. Kun vertailtiin MQTT-protokollaa ja HTTP-protokollaa 3G-verkossa, MQTT oli 93 kertaa nopeampi. [16.]

4.3 Botti

Bottien tarkoitus on avustaa käyttäjiä yksinkertaisissa tehtävissä, kuten lennon varauksissa tai tiedon etsinnässä. Botteja voidaan ajatella tietokoneohjelmina, jotka pystyvät palvelemaan monia ihmisiä samanaikaisesti. Vaikka botti ei ole yhtä joustava kuin ihminen, on kuitenkin tilanteita, joissa botin käyttäminen on tehokkaampaa kuin ihmisen palkkaaminen samaan tehtävään. [17.]

Jotta palvelin voisi ilmoittaa rajan ylityksestä henkilölle, on puhelimessa oltava kommunikointitapa, joka ei ole riippuvainen kommunikoinnista tekstiviestien kautta. Tähän tarkoitukseen valittiin Telegram-niminen keskustelupalvelu. Telegrammia käytettiin projektissa henkilön ja palvelimen välisenä kommunikaatioväylänä. Telegram valittiin projektiin, koska sillä botin luominen palveluun on todella helppoa. Botti luodaan puhumalla ”BotFather”-nimiselle botille. Botin luomisen saa tehtyä yhdellä komennolla ja antamalla kaksi eri nimeä botille. Tämän jälkeen tarvitaan HTTP-API (Application Programming Interface) -merkkijono. Tällä merkkijonolla voidaan bottia kutsua esimerkiksi Python-sovelluksesta käyttämällä Telegramin omia kirjastoja. [18.]

Kun henkilö astuu turva-alueen ulkopuolelle, lähetetään hänelle Telegramiin viesti ”Return back” eli palaa takaisin. Tässä vaiheessa palvelin ottaa ajan ylös ja lisää siihen 30 sekuntia. Kun 30 sekuntia on mennyt eikä henkilö ole mennyt takaisin turva-alueelle, palvelin lähettää henkilölle toisen viestin: ”Drone has been sent” eli drooni on lähetetty. Tässä vaiheessa droonille on lähetetty herätysviesti. Drooni lähetetään liikkeelle tarkistamaan henkilön sijainti, vaikka henkilö palaisi tässä välissä alueelle takaisin. Vaikka nämä asiat olisivat tapahtuneet, henkilön palatessa takaisin alueelle hänelle lähetetään viesti ”Thank you for returning back” eli kiitos, että palasit takaisin. Kuvassa 15 näkyy, minkälaiselta Telegram näyttää, kun viestejä on lähetetty testien aikana.



Kuva 15. Esimerkki Python-sovelluksen lähettämistä viesteistä Telegramissa.

4.4 Drooni

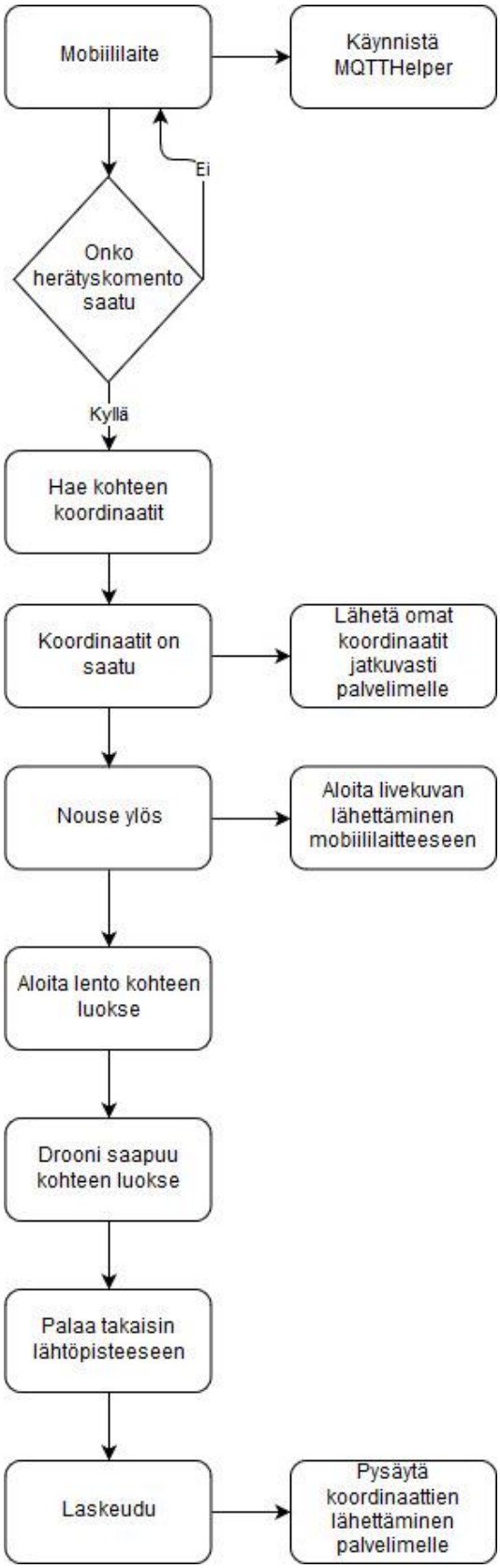
Drooni on teknisesti miehittämätön ilma-alus. Droonien yleinen kutsumanimi ympäri maailmaa on englanninkielinen nimi ”drone” tai UAV (Unmanned Aerial Vehicles). Droonit ovat tavallaan lentäviä robotteja, sillä ne voivat lentää ilman, että ihminen ohjaa niitä kauko-ohjaimella tai muulla tavalla. Drooneja myös kuitenkin voi lennättää mm. kauko-ohjaimella. Ilman ihmisen ohjausta on tärkeää, että droonissa on sensoreita, jotka auttavat sitä väistelemään esteitä, ja GPS, joka ohjaa droonia lentämään oikeaan paikkaan. [19.]

Lähihistoriassa drooneja on käytetty sotilaskäytössä, mutta nykyisin ne ovat laajasti siviilikäytössä. Markkinoille on tullut myös paljon drooneja harrastekäyttöön. Huomioon on tullut myös monia tapauksia ja huolia, joissa drooneja on käytetty ihmisten vaikoilussa tai paparazzikäytössä. [19.] Jos insinöörityön prototyyppiä jatkokehitetään, on otettava huomioon lakiasiat. Koska droonit voivat lentää korkealle, on varottava yhteentörmäyksiä lentokoneitten kanssa. Droonien pitää lentää alueilla, joissa se ei voi myöskään vahingoittaa ihmisiä esimerkiksi putoamalla heidän päällensä.

Droonin mukana tulleen ohjaimen kautta voidaan lähettää signaaleja enimmillään kahden kilometrin päähän [20]. Tämän kantomatkan pitäisi olla riittävä prototyypin tarkoituksiin. Drooni vastaanottaa komentoja ohjaimesta radioaaltojen avulla ja ohjain vastaanottaa komentoja mobiililaitteesta. Hyvän kuvanlaadun vuoksi mobiililaite eli puhelin kannattaa olla kiinnitettynä ohjaimessa olevassa puhelinpidikkeessä. Kuvassa 16 näkyy DJI Spark -drooni.

Kuva 16. DJI Spark -drooni [20].

Drooni ei ota komentoja vastaan suoraan mobiililaitteesta, vaan sitä ohjataan droonin omalla ohjaimella. Ohjain on yhteydessä mobiililaitteeseen Wi-Fi-yhteydellä. Ohjain siis toimii välittäjänä mobiililaitteen ja droonin välillä. Mobiililaite oli prototyypissä Android-puhelin, joka vastaanottaa komentoja palvelimelta MQTT-protokollan avulla. Kun droonia ohjaava Android-sovellus saa herätyskomennon palvelimesta, käynnistyy prosessi, joka nappaa MQTT-palvelimesta henkilön koordinaatit ja aloittaa lennon. Ennen kuin drooni aloittaa lennon, se hakee omat koordinaattinsa, jotta se voi palata lennon jälkeen takaisin. Tämän jälkeen drooni aloittaa lennon lentämällä henkilön koordinaatteihin. Koko lennon ajan drooni lähettää livekuvaa takaisin päätelaitteeseen. Saavuttuaan kohdekoordinaatteihin se palaa se takaisin kotikoordinaatteihinsa. Kuvassa 17 näkyy droonin toimintaperiaate kaaviona.



Kuva 17. Droonin toimintatapakaavio.

DJI Spark -drooni valittiin prototyyppiin, kun laitetta tutkiessa havaittiin, että siinä oli kaikki tarvittavat ominaisuudet. Myös hinnaltaan tämä drooni oli sopiva, sillä projektissa drooni oli omakustanteinen. Drooni vaikutti lupaavalta, sillä siinä on sisäinen GPS-paikannin ja ohjain, joka pystyy pitämään yhteyden drooniin enintään kaksi kilometrin alueelle. Jokaisen akun 16 minuutin lentoaika ja pieni koko tekivät siitä helposti siirrettävän. Spark-droonissa on myös 1920 x 1080 pikselin kamera, joka on enemmän kuin tarpeeksi prototyyppiin. [20.] Myös toisen DJI-droonin ostamista Sparkin sijaan harkittiin. DJI Mavic Pro on DJI-kaupassa hinnaltaan kaksi kertaa kalliimpi kuin DJI Spark. Mavic Pron ominaisuuksiin kuuluu mm. 27 minuutin lentoaika ja 7 kilometrin ohjausalue, mutta korkeamman hinnan takia valittiin Spark-drooni. [21.]

Droonin heikkoudet

Projektissa oli tarkoitus käyttää droonin sisäänrakennettuja komentoja, joilla droonin saa lennettyä tiettyyn paikkaan, mutta Spark-mallissa ei ole mahdollista käyttää sitä. Jotta droonin saisi menemään oikeaan paikkaan ilman ihmisen ohjausta, oli pakko käyttää toiseen tarkoitukseen tarkoitettua ”go home” -toimintaa. Tämän toiminnan käyttäminen vaati uusien kotikoordinaattien asettamista aina, kun droonin oli tarkoitus lentää uuteen kohteeseen.

Kenttätestaukset oli tehtävä useissa lyhyissä testilennoissa, jotta drooni tai sen akku ei kärsisi liikaa. Myös kovaa tuulta tuli välttää, sillä ongelmien tullessa olisi vaikeaa tietää, johtuivatko ongelmat ohjelmistosta vai tuulesta tai muista säähän liittyvistä syistä. Osa drooniin liittyvistä ongelmista voi olla peräisin myös heikosta GPS-yhteydestä. Jos droonilla on liian vähän yhteyksiä satelliitteihin, ei drooni luota koordinaatteihin ja virheiden välttämisen takia se ei lähde lentämään.

Myös Wi-Fin käyttö droonin ja ohjaavan puhelimen yhteydessä oli ongelma, sillä puhelimet oletuksena käyttävät vain yhtä yhteyttä internetin käytössä. Tavallisesti puhelin käyttää ensisijaisesti Wi-Fi-yhteyttä. Puhelin ei voinut olla yhteydessä mobiilidataverkkoon, jos se oli yhteydessä droonin ohjaimeen Wi-Fi-yhteydessä. Tämän ongelman välttämiseksi käytettiin kolmannen osapuolen ohjelmaa ”Speedify”, jonka ansiosta pystyttiin pitämään yhteyttä Wi-Fillä drooniin ja mobiilidatalla verkkoon.

5 Kehitysmahdollisuudet

Kun insinöörityön prototyyppi saatiin valmiiksi, tuli paljon jatkokehitysideoita. Vaikka järjestelmä toimii moitteettomasti, on aina tilaa jatkokehitykselle. Ensimmäinen jatkokehitysvaihe olisi todennäköisesti kehittää järjestelmä tukemaan useaa henkilöä. Jatkokehitysideoita olisi mahdollista tutkia ja kehittää, jos prototyyppi todetaan käyttökelpoiseksi. On myös mahdollista kehittää järjestelmää toisiin käyttötarkoituksiin, joissa tarvitaan samantyyppisiä ominaisuuksia. Järjestelmästä voisi myös tehdä vartiointijärjestelmän, johon lähetetään koordinaatteja hälytyksinä, ja drooni lähetettäisiin paikanpäälle. Tämän tekeminen olisi mahdollista vaihtamalla paikannusjärjestelmät staattisiin hälyttimiin ja muokkaamalla alueet tärkeyden mukaan.

5.1 Paikannusjärjestelmä

Prototyypissä käytettyä sovellusta olisi helppoa jatkokehittää tukemaan useampaa puhelinta. Olisi mahdollista, että käyttäjä voisi itse asettaa esimerkiksi oman nimensä tai jokaiselle henkilölle olisi asetettu oma nimimerkkinsä. Muut järjestelemät tunnistaisivat koordinaattien lähteen nimimerkkien avulla. Olisi myös tärkeää muuttaa koordinaattien lähetystä niin, että sovellus lähettää henkilön sijaintia, vaikka henkilö ei liikkuisikaan. Projektissa päätettiin käyttää vain liikkuvan puhelimen koordinaatteja akun virran säästämiseksi.

Projektissa käytetty puhelin valittiin, koska se oli valmiiksi saatavilla, sen yhteysmahdollisuudet ovat hyvät ja siinä oli sisäänrakennettu GPS. Tästä seuraava askel olisi tehdä mikrokontrollerista ja muista piireistä laite, joka pystyy lähettämään sijaintiaan samalla tavalla kuin prototyyppiin tehty sovellus. Tähän liittyviä haasteita on monia, kuten riittävän tehokkaan akun löytäminen ja yhteyden muodostaminen.

5.2 Palvelin

Palvelimen jatkokehityksessä olisi tärkeää, että useamman henkilön koordinaatteja voitaisiin tarkkailla. Nykyiset ohjelmat voitaisiin melko helposti muuntaa tukemaan montaa paikannusjärjestelmää samanaikaisesti. Tämän projektin aikana oli tarkoitus todistaa konseptin toimivuus, joten keskityttiin vain yhden paikannusjärjestelmän tukemiseen.

Projektin loppupuolella palvelin tarkistaa koordinaatit aina, kun uusi koordinaatti saapuu MQTT-palvelimelle. Olisi mahdollista jatkokehittää järjestelmä tarkistamaan koordinaatit esimerkiksi sekunnin välein. Toiminnan kannalta tällä on vähän vaikutusta, sillä uusia koordinaatteja olisi tarkoitus tulla noin joka toinen sekunti. Nykyisen järjestelmän parempi puoli on, että drooni herätetään vain, jos koordinaattien tiedetään olevan alueen ulkopuolella. Ajoitettu järjestelmä luottaisi siihen, että henkilö pysyy paikallaan, jos koordinaattien tulo lakkaa hetkeksi.

Verkkosivun seuraava aste olisi tehdä droonin ja henkilöiden merkkien liikuttamisesta sulavampaa. Projektin aikana merkit siirretään uusiin koordinaatteihin kartalla. Sulavampi liikuttaminen tekisi kartasta käyttäjäystävällisemmän. Kameran sijainnin siirtäminen alueen yläpuolelle tekisi kartasta huomattavasti käyttäjäystävällisemmän, jos turva-alue siirtyisi usein.

Projektissa käytetty CloudMQTT-pilvipalvelu on hyödyllinen ja helppo tapa lähettää dataa laitteista palvelimelle, mutta parannusta tarvitaan, jos järjestelmästä halutaan tehdä monia ihmisiä tukeva. CloudMQTT tarjoaa ilmaiseksi 10 laitteen kommunikaation välittäjälle. Tämä tarkoittaa, että enintään 10 laitetta voi kommunikoida välittäjälle samaan aikaan. On mahdollista, että CloudMQTT-palvelusta maksettaisiin ja näin saataisiin lisättyä tuettujen laitteitten määrää. Toisina vaihtoehtoina olisi vaihtaa toiseen palveluun tai rakentaa oma MQTT-välittäjä.

5.3 Kommunikaatio palvelimesta henkilölle ja drooni

Prototyypissä käytettiin Telegram-nimistä viestinlähettämissovellusta. Tämä sovellus tarvitsee internetyhteyden, joten henkilö ei voi saada viestejä, ellei ole saatavilla internetyhteyttä. Jatkokehityksessä olisi mahdollista asentaa SIM-kortilla toimiva moduuli, joka sallisi viestien lähettämisen tekstiviestinä. Tämä sallisi järjestelmän käytön paikoissa, joissa ei ole vahvaa mobiilidatasignaalia.

Myös hoitokodin henkilökunnalle voitaisiin lähettää viesti, jos hoitokodin asukas liikkuu alueen ulkopuolella. Näin henkilökunta pystyy reagoimaan nopeammin ja mahdollisesti välttämään vaaratilanteita. Yksi toteutusvaihtoehto on laittaa turva-alueita lähiseudun alueelle ja näin mahdollistaa turvallisemman liikkumisen lähiseudulla. Tämäntyyppinen levitetty turva-alue voi mahdollisesti olla kätevä, jos henkilökunnan jäsen käy lähiseudulla ja hänellä on mukanaan asukkaita. Henkilökunnan jäsenelle tulisi hälytys ja asukkaan olinpaikka, jos asukas menee alueen ulkopuolelle.

Prototyypissä käytetty DJI Spark -drooni on tarkoitettu kuluttajakäyttöön. Tämä drooni oli hinnaltaan huomattavasti halvempi kuin isommat ja tehokkaammat droonit. Jos haluttaisiin drooni, joka kestäisi rankemmankin tuulen ja pystyisi lähettämään livekuvaa selvemmin kuin Spark, olisi hyvä sijoittaa kalliimpaan laitteeseen. Kalliimmat droonit tarjoaisivat suuremman toimivuusalueen, paremman akkukeston ja selvemmän kuvan. DJI:n kalliimmissa drooneissa on dokumenttien mukaan myös tarvittavia ominaisuuksia koodimielessä. Vaikka Spark-drooni saatiin toimimaan prototyypissä, olisi kalliimmilla drooneilla kehittäminen ollut helpompaa.

6 Yhteenveto

Insinöörityössä oli tavoitteena rakentaa järjestelmä, joka pystyisi paikantamaan henkilön, lähettämään henkilön koordinaatit prosessoitavaksi ja tarpeen tullen lähettämään viestin drooniin. Työhön oli asetettu myös ylimääräiseksi tavoitteeksi saada valmiiksi graafinen käyttöliittymä, josta voitaisiin nähdä laitteiden sijainnit ja turva-alue sekä mahdollisesti asettaa uusi turva-alue.

Tuloksena saatiin valmiiksi toimiva järjestelmä, jolla pystytään havaitsemaan tietyn laitteen menevän yli alueen rajasta. Tämä alue voidaan asettaa tarpeen tullen uudelleen käyttäen samaa laitetta, jolla henkilöä paikannetaan. Vaikka graafisessa käyttöliittymässä eli verkkosivussa ei voi asettaa uutta turva-aluetta, siinä on kuitenkin mahdollista nähdä laitteiden ja alueen sijainnit kartalla.

Parannettavaa olisi paikannusjärjestelmän kehittämisessä pienemmäksi laitteeksi. Työssä käytetty puhelin on kätevä prototyypin kehittämiseen tai henkilökohtaisena paikantimena. Jos paikannuslaitetta kehitettäisiin seuraavaan asteeseen, olisi halvempi ja kompaktimpi laite parempi. Verkkosivua tulisi kehittää graafiseksi käyttöliittymäksi, jossa pystyisi hallitsemaan alueita tehokkaammin. Projektissa uudet alueet asetettiin paikantimella. Jos verkkosivu toimisi käyttöliittymänä alueitten asettamisessa, voisi olla mahdollista lisätä muitakin kuin ympyrän muotoisia alueita. Myös useamman alueen asettaminen olisi mahdollista. Itse palvelin toimii hyvin työn aikana, mutta palvelimesta tulisi tehdä montaa henkilöä tukeva järjestelmä. Tämä olisi suhteellisen helppoa, sillä ohjelma on suunniteltu joustavaksi.

Lähteet

1 Kulonen, Sirpa. 2013. Älä unohda muistisairasta! Verkkoaineisto. <http://www.potilaanlaakarilehti.fi/uutiset/ala-unohda-muistisairasta/> Luettu 13.4.2018.

2 Vuorio, Jukka. 2016. Muistisairas 84-vuotias Anja katosi talviyöhön ja menehtyi – miten näin pääsi tapahtumaan? Verkkoaineisto. <https://seura.fi/asiat/tutkitut/p305793/> Luettu 13.4.2018.

3 Muistisairauteen liittyviä turvallisuusriskejä. 2017. Verkkoaineisto. Muistiliitto. < https://www.muistiliitto.fi/fi/muistisairaudet/erityiskysymyksia/muistiystavallinen-ymparisto-turvallisuus/turvallisuusriskeja> Luettu 13.4.2018.

4 Helle on iso riski iäkkäille – mitä voisimme tehdä. 2017. Verkkoaineisto. Suomen sydänliitto ry. < https://sydan.fi/terveys-ja-hyvinvointi/helle-iso-riski-iakkaille-mita-voisimme-tehda> Luettu 13.4.2018.

5 Eclipse Paho. 2018. Verkkoaineisto. Eclipse Foundation. <https://www.eclipse.org/paho/> Luettu 5.4.2018.

6 Eclipse Paho. 2018. Verkkoaineisto. Eclipse Foundation. < https://www.eclipse.org/paho/clients/android/> Luettu 5.4.2018.

7 Mitchell, Bradley. 2018. Servers are the heart and lungs of the Internet. Verkkoaineisto. <https://www.lifewire.com/servers-in-computer-networking-817380> Luettu 11.4.2018.

8 McCarty, Bill. 2005. Linux: Fedora & Red hat enterprise Linux. Helsinki: Readme.fi.

9 Oksanen, Lasse. 2018. Raspberry Pi -Tiedostopalvelin ja Web-käyttöliittymä. Verkkoaineisto <http://www.theseus.fi/bitstream/handle/10024/141886/Oksanen_Lasse.pdf?sequence=1&isAllowed=y> Luettu 10.4.2018.

10 Campbell, Alex. 2017. How to have a Linux home server on the cheap. Verkkoaineisto. <https://www.pcworld.com/article/3184925/linux/how-to-have-a-linux-home-server-on-the-cheap.html> Luettu 10.4.2018.

11 About Apache. 2018. Verkkoaineisto. The Apache Software Foundation. <https://httpd.apache.org/ABOUT_APACHE.html> Luettu 12.4.2018.

12 Robbins, Jennifer Niederst. 2006. Web design in a nutshell: a desktop quick reference. O'Reilly Media Inc.

13 Setting up an Apache web server on Raspberry Pi. 2018. Verkkoaineisto. Raspberry Pi Foundation. <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md> Luettu 12.4.2018.

14 CloudMQTT Documentation. 2018. Verkkoaineisto. CloudMQTT. <https://www.cloudmqtt.com/docs.html> Luettu 9.4.2018.

15 CloudMQTT Plans. 2018. Verkkoaineisto. CloudMQTT. <https://www.cloudmqtt.com/plans.html> Luettu 12.4.2018.

16 Serozhenko, Marina. 2017. MQTT vs. HTTP: which one is the best for IoT? Verkkoaineisto. <https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105> Luettu 9.4.2018.

17 Palo, Jussi. 2017. Botit yritysmaailmassa – tätä päivää vai tulevaisuutta? Verkkoaineisto. < https://www.sulava.com/botit-yritysmaailmassa/> Luettu 8.4.2018.

18 Bots: An introduction for developers. 2018. Verkkoaineisto. Telegram. <https://core.telegram.org/bots> Luettu 8.4.2018.

19 Rouse, Margaret. 2016. Drone (Unmanned Aerial Vehicle, UAV). Verkkoaineisto. <https://internetofthingsagenda.techtarget.com/definition/drone> Luettu 9.4.2018.

20 DJI Store: Spark. 2018. Verkkoaineisto. DJI. <https://store.dji.com/product/spark?site=brandsite&from=buy_now_bottom&vid=24641> Luettu 2.4.2018.

21 DJI Store: Mavic pro. 2018. Verkkoaineisto. DJI. <https://store.dji.com/product/mavic-pro?site=brandsite&from=buy_now_bar> Luettu 12.4.2018.